DARPA Urban Challenge Technical Paper

Team Leader: Charles Reinholtz
Team Name: ***V***ictor***T***ango
Team Reference Number: A111
Submission Date: 13 April 2007

Corresponding Author:
Charles Reinholtz
Alumni Distinguished Professor
Randolph Hall, Mail Code 0218
Virginia Tech
Blacksburg, VA 24061
creinhol@vt.edu

Additional Authors:

Thomas Alberi	Scott Frash
David Anderson	Grant Gothing
Andrew Bacha	Jesse Hurdus
Cheryl Bauman	Shawn Kimmel
Stephen Cacciola	Chris Sharkey
Patrick Currier	Andrew Taylor
Aaron Dalton	Chris Terwelp
Jesse Farmer	David Van Covern
Ruel Faruque	Mike Webster
Michael Fleming	Al Wicks

Table of Contents

Executive Summary

The DARPA Urban Challenge offers an incredible opportunity to advance the state of the art in autonomous vehicle technology. Team VictorTango is acutely aware of this opportunity; our goal is to develop an autonomous vehicle system that will revolutionize unmanned systems.

Our commitment to excellence begins with our vehicle platform, Odin, a 2005 Ford Escape Hybrid. Sensors, computers, power systems, and control elements have been integrated into Odin, resulting in an autonomous platform providing an extraordinary combination of safety, reliability, and ease of operation.

Extensive effort has gone into the selection and location of sensors and into the development of algorithms and software. Data from multiple sensors is fused to perceive road coverage and to detect and differentiate static obstacles from vehicles. Where possible, we have relied on proven navigation approaches, but we have also developed revolutionary new approaches to ensure a successful progression from Basic Navigation through Advanced Traffic requirements. Important innovations include the development of novel software architectures, the creation of a custom Urban Challenge simulator, and total commitment to JAUS to enable cross-platform compatibility. Most importantly, the work of team VictorTango is bringing software out of simulation and into the field.

1. Introduction

On November 3rd, 2007, DARPA will host the Urban Challenge, an autonomous ground vehicle race in an urban environment. To meet this challenge, Virginia Tech and TORC Technologies have formed team VictorTango and have developed Odin, a 2005 Ford Hybrid Escape modified for autonomous operation. Team VictorTango is a collaborative effort between academia and industry. The team includes 46 undergraduate students, 8 graduate students, 4 faculty members, 5 full time TORC employees and industry partners, including Ford Motor Co. and Caterpillar, Inc.

The DARPA Urban Challenge requires an autonomous ground vehicle to navigate an ordered list of checkpoints in a road network. The road network is supplied as a-priori information in the form of a Route Network Definition File (RNDF), listing each road and contained lanes as well as regions called zones used to represent parking lots or other unstructured environments. The RNDF describes each lane by a series of waypoints and flags certain waypoints as entrances and exits that connect each lane to the road network. A series of Mission Data Files (MDF) are provided that specify checkpoints which must be visited in a specified order. The vehicle must begin operation less than five minutes after receipt of the MDF. The vehicle must choose roads considering road speed limits, possible road blockages and traffic conditions to achieve the checkpoints as fast as possible.

The vehicle must adhere to rules of the road, specifically California state driving law, as well as DARPA mandated rules, such as vehicle separation distances. When traveling, the vehicle must remain centered in the travel lane and react safely to other vehicles by matching speeds or passing when appropriate. Intersections are defined in the RNDF using waypoints specified as exits or entrances and are optionally marked as a stop indicating a stop sign behavior. The vehicle is not required to sense any signs or signals such as traffic lights, however the vehicle must still obey right-of-way rules and must proceed in the correct order when at a stop point. The vehicle must drive safely and defensively, avoiding both static and dynamic obstacles at speeds of up to 30 mph and be able to account for and predict the future movement of the other vehicles on the road.

2. Overview

Team ***V***ictor***T***ango has divided the problem posed by the Urban Challenge into three major parts: perception, planning, and base vehicle platform.

2.1 Perception Architecture Overview

To fulfill the behavioral requirements of the Urban Challenge, Odin must first be able to adequately localize its position and perceive the surrounding environment. Since there may be sparse waypoints in the RNDF and areas of poor GPS coverage, the surrounding road coverage and legal lanes of travel must also be sensed. Finally, Odin must be able to perceive all obstacles in its path and appropriately classify obstacles as vehicles.

For each perception requirement, multiple sensors are desirable to achieve the highest levels of fidelity and reliability. To allow for maximum flexibility in sensor fusion, the planning software does not use any raw sensor data; rather it uses a set of sensor-independent perception messages. The perception components and the resulting messages are shown in Figure 1. The Localization component determines the vehicle position and orientation in the world. The Road Detection component determines a road coverage map as well as the position of each lane in nearby segments. The Object Classification component detects obstacles and classifies them as either static or dynamic. A dynamic obstacle is any obstacle that is capable of movement, so a stopped vehicle would be classified as a dynamic obstacle with zero forward velocity.

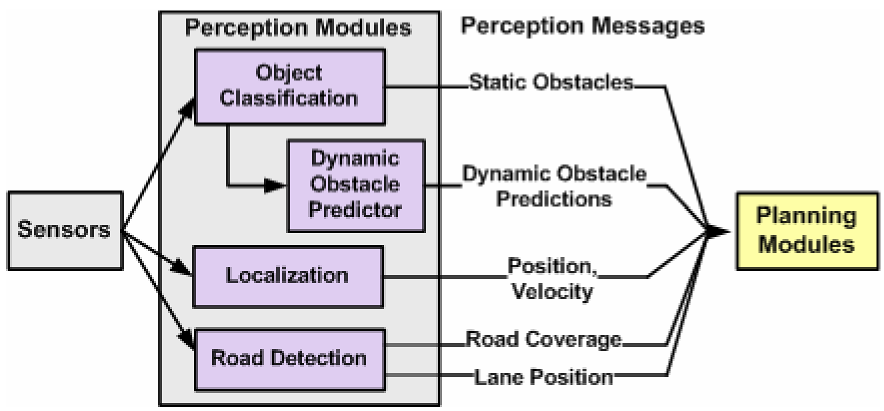


Figure 1: Perception structure overview

2.2 Planning Architecture Overview

The planning software on Odin uses a Hybrid Deliberative-Reactive model dividing upper level decisions and lower level reactions into separate components. These components run at independent rates, allowing the vehicle to react to emergency situations without needing to re-plan an entire route. Splitting the decision making into separate components allows each system to be tested independently. It also allows for parallel development, which is especially attractive given the short development timeline of the DARPA Urban Challenge. An overview of the planning process is shown in Figure 2.

The Route Planner component is the coarsest level of planning and is responsible for determining which road segments and zones the vehicle should use to travel to all checkpoints. It will update the desired route upon request in situations such as blocked segments or traveling behind slow traffic.

The Driving Behaviors component is responsible for obeying the rules of the road and guiding the vehicle along the planned route. This includes deciding when lane changes are necessary, determining the progression order at intersections and navigating zones. To maintain these behaviors, situational awareness and high-level decision making must be achieved. The Driving Behaviors component continuously determines a set of goal points and motion guidelines for the Motion Planner in the form of a behavior profile. This behavior profile takes into account object, road, and route-network information and works to achieve the optimal route provided by the Route Planner.

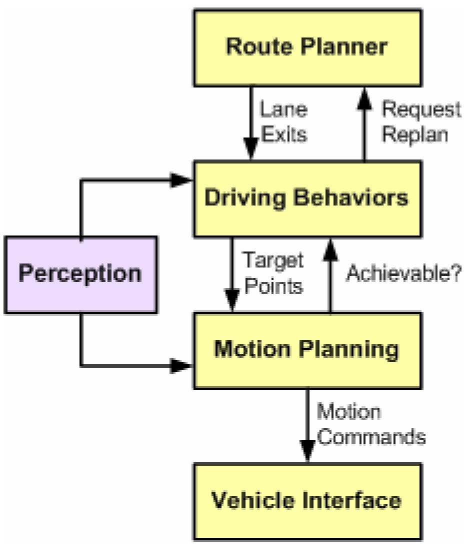


Figure 2: Planning structure overview.

The lowest level of the planning process is the Motion Planning component. Motion Planning receives behavior profiles and plans the proper sequence of vehicle commands to achieve the desired behavior. The motion planner uses the dynamic constraints of the vehicle, all perceived road data, and both static and dynamic obstacle data to plan a safe trajectory. If a behavior profile cannot be achieved, it is reported back to the Driving Behaviors component. The output of the Motion Planner is a sequence of curvatures and speeds that the Vehicle Interface will translate to the necessary vehicle actuators. The curvature-based output of the Motion Planner allows the software to remain platform independent. Only the Vehicle Interface component changes when moving software between vehicle platforms. This has allowed use of the Virginia Tech Grand Challenge vehicles (Cliff & Rocky) and autonomous Cadillac SRX for testing and development.

2.3 Base Platform Overview

Team ***V***ictor***T***ango's entry in the Urban Challenge is a modified 2005 Hybrid Ford Escape named Odin, shown in Figure 3. This base vehicle platform meets the requirement of a midsize commercial automobile with a proven safety record. The use of the hybrid-electric Ford Escape provides numerous advantages in the areas of on-board power generation, reliability, safety and autonomous operation. As required by DARPA, the drive-by-wire conversion does not bypass any of the stock safety systems. Since the stock steering, shifting and throttle systems on the Hybrid Escape are already drive-by-wire, these systems can be controlled electronically by emulating the command signals,

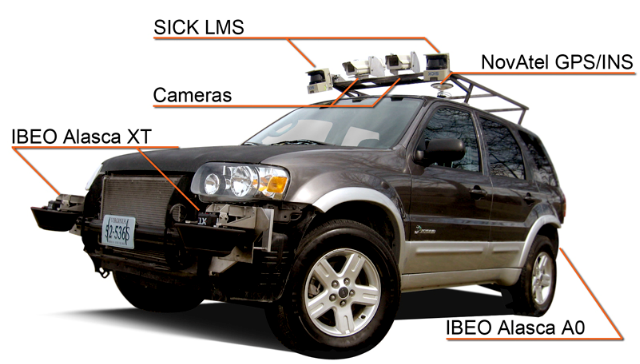


Figure 3: External view of Odin with sensors labeled.

eliminating the complexity and failure potential associated with the addition of external actuators. The stock hybrid power system is able to provide sufficient power for sensors and computers without the need for a separate generator.

3. Analysis & Design

This section presents the justification of the major design choices made in the development of Odin. For clarity, the section is organized into three major parts: perception, decision making, and base vehicle platform. In each of these sections, the design requirements are reviewed, the specific implementation is discussed, and the justification for decisions is provided.

3.1 System Architecture & Communications

Previous Grand Challenges could be solved using a purely Reactive software architecture; however the complex nature of the Urban Challenge necessitates a hybrid solution. In addition to the simpler goal-seeking behavior required in the previous challenges, Urban Challenge vehicles must maintain knowledge of intent, precedence, and timing. With many concurrent perception and planning tasks of varying complexity, priority, and computation time, parallelism is preferred to a single monolithic Sense-Plan-Act structure [Murphy, 2000].

Victor***T***ango's software structure employs a novel Hybrid Deliberative-Reactive paradigm. Odin's perception, planning, and acting occurs at several levels, in parallel tasks running at different refresh rates, acting on the most recent information received from other modules. An overview of the hybrid mixture of deliberative planning, reactive navigation, and concurrent sensor processing is shown in Figure 4.

The primary design goals for Odin's communications architecture include support for parallelism and the Hybrid Reactive-Deliberative paradigm; portability between different vehicle platforms, which is especially important for testing; modularity, which eases the division of work in a large team; and automated configuration, which allows software elements to be moved between computers without reconfiguration. JAUS (the Joint Architecture for Unmanned Systems) was implemented for communications, because it provides support for all of these design goals. Each software module is implemented as a JAUS component running on one of the computing nodes. All interactions between software modules occur via JAUS messages.

The most useful features of JAUS are the message routing and dynamic configuration capabilities, which allow components to be moved from one computing node to another without changing settings. In addition to the standard JAUS messages, approximately 60 additional experimental messages were implemented for the Urban Challenge to pass data such as Road Coverage and Lane Position.

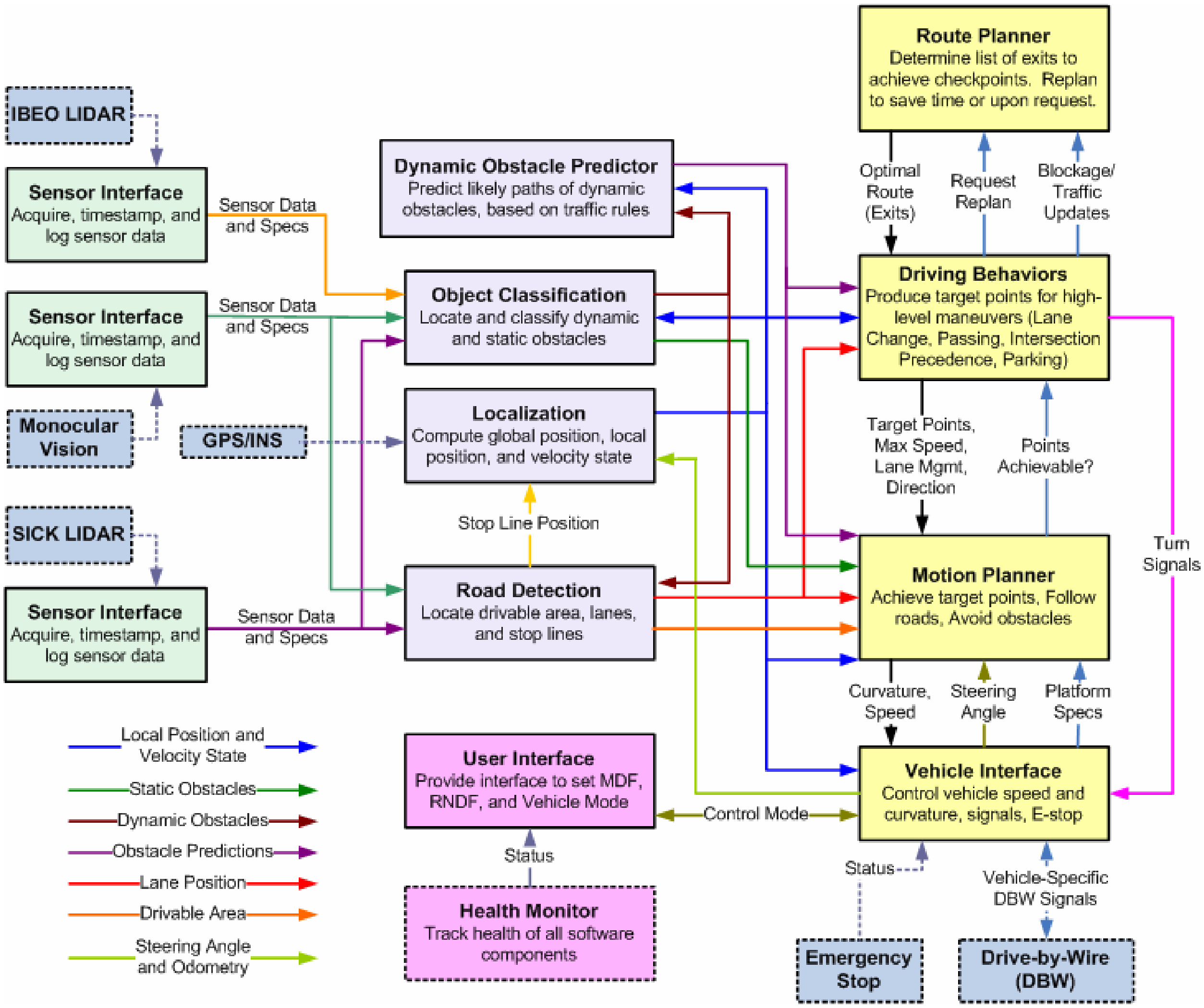


Figure 4: System Architecture for Odin. All interaction between software modules takes place via JAUS messages. For clarity, the connections from all software elements to the Health Monitor are omitted.

3.2 Perception

Perception is defined to include all aspects of the design necessary to sense the environment and transform the raw data into information useful to the decision making software.

3.2.1 Sensor Selection

Odin's sensor coverage is dictated by the fields of view and the ranges needed to safely and successfully navigate through an urban environment at speeds up to 30 mph. The first coverage area considered was the area in front of the vehicle. Obstacles detected in this area pose the greatest threat for a collision, so it is important that the vehicle is able to sense these objects early enough to react to them. From the Urban Challenge rules, the maximum speed that a vehicle will be traveling within the competition course is 30 mph, yielding a maximum differential speed between Odin and a dynamic obstacle of 60 mph. Testing has shown that Odin's steering and braking systems require approximately 1.25 seconds to safely initiate and complete an evasive maneuver in a worst-case, high-speed head-on collision scenario. From these tests, it has been calculated that the vehicle's forward-looking sensors must have a range of at least 50 meters to appropriately react in this extreme case.

The sensor suite must also have a field of view that will allow Odin to detect dynamic obstacles all around the vehicle. This field of view is necessary for the vehicle to appropriately react to other vehicles approaching from the side or traveling in adjacent lanes. It is also necessary for the vehicle to travel in reverse. Since Odin will move slowly in reverse, and cannot slide laterally, the maximum relative velocity between Odin and a vehicle approaching from the side or rear will be 30 mph. The range necessary for sensing obstacles to the side and rear of the vehicle was thus taken to be half of the forward-looking distance, or 25 meters.

Odin's sensor systems must also be able to classify the objects it sees. All objects will be identified as either static or dynamic obstacles. From the Urban Challenge rules, the only dynamic obstacles on the competition course will be vehicles (i.e no pedestrians). Any obstacle having a significant velocity and size is, therefore, classified as a vehicle. The sensor system must also have a means of identifying stationary vehicles and classifying them as dynamic objects with zero velocity. This is particularly important in cases such as vehicles being queued at an intersection, or temporarily disabled.

Another requirement of the sensor system is the ability to detect roads, drivable areas, lane markings, and stop lines in front of the vehicle. Odin needs to be aware of road coverage and lane markings in the lane in which it is traveling as well as adjacent lanes. Since the average width of a travel lane is 3.6m, the required sensor field of view must be at least 11m in width (3 lane widths) in front of Odin. This enables Odin to detect adjacent lanes on either side.

Odin's sensor suite must also provide an absolute measurement of its position and orientation on the Urban Challenge course. Tasks such as stopping at stop lines and parking will require precise navigation, which requires precise localization. Odin's Localization module will combine inputs from several sensors to provide the best possible position solution. However, this solution is only guaranteed to be as accurate as the most accurate absolute measurement. When geo-referenced landmarks are not visible, this absolute measurement can only be provided by a Global Positioning System (GPS). Therefore, using as accurate a GPS system as possible will improve the overall Localization performance. For this reason, the GPS accuracy requirement has been established to be less than 1 meter CEP.

The GPS/INS solution chosen for Odin is a NovAtel Propak LB+ system with Omnistar HP differential corrections. The Propak LB+ system uses NovAtel's SPAN technology to fuse IMU and GPS data to provide a smooth 3D position, velocity, and attitude solution. This system will provide Odin with a full position solution with an accuracy of 0.1 meters CEP, even in areas with poor GPS satellite coverage.

3.2.2 Sensor Layout

The sensor coverage for Odin is shown in Figure 5. The largest portion of Odin's detection coverage is provided by a coordinated pair of IBEO Alasca XT Fusion laser rangefinders. This system includes two 4-plane rangefinders and a single external control unit (ECU) that covers a 260 degree field of view, as shown in Figure 5. The system has an advertised range of almost

200 meters, although the effective range to reliably identify most objects has been shown in testing to be closer to 100 meters. IBEO rangefinders also utilize multi-return technology, filtering out the effects of dust or precipitation, which could otherwise return false readings. A single IBEO Alasca A0 unit with a range of 80 meters and a field of view of 150 degrees is used to detect vehicles behind Odin and navigate in reverse.

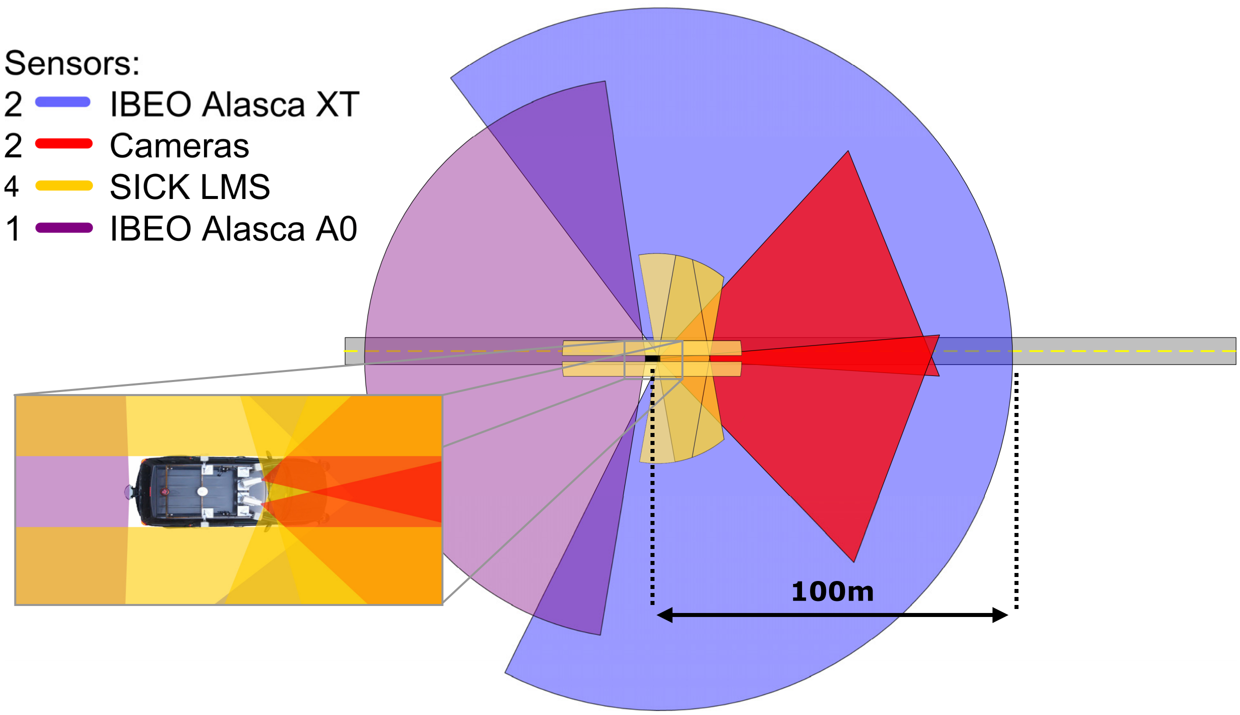


Figure 5: Odin's sensor coverage. The colored areas indicate the maximum range of the sensor or the point at which the sensors scanning plane intersects the ground. Odin is facing to the right in this figure.

The ECUs for the XT Fusion and A0 each contain commercially available object classification software that is able to group scan points together to form discrete objects with known positions, velocities, and sizes. Tests have shown that the software's segmentation is accurate, especially for moving vehicles. The classification software does have difficulty with multiple small objects closer than 1 meter apart, which are often grouped together as a single larger object. Also, the classifications for stationary objects are often incorrect. These inconsistencies have been remedied with a custom developed post-processing filter that acts on object age, velocity, and acceleration.

Two Imaging Source color monocular cameras are used to supplement the IBEO classification software. In combination, the cameras cover a 90-degree horizontal field of view in front of Odin, and they are capable of simultaneously transmitting two raw 1024 by 768 images at a rate of 15 frames per second across an IEEE 1394 connection. A number of automatic adjustment features such as shutter speed and gain are available, and other features such as exposure time are controlled in software to maintain consistent lighting conditions. These cameras are also used as the primary means of road detection. In testing, visual information such as color and texture has proven to be more effective than laser scanners in detecting roads.

It is difficult to detect negative obstacles with monocular vision processing alone. Since the IBEO rangefinders are oriented such that they do not intersect the road on flat ground, two SICK LMS 291s are mounted on the front corners of the roof rack, angled downward in order to detect sharp changes in the otherwise level road. In addition, two side-mounted SICK LMS 291 single plane rangefinders are used as simple “bumpers” to cover the side blind spots of the vehicle and ensure 360-degree coverage.

3.2.3 Road Detection

To successfully complete missions, Odin must be able to identify roads and road boundaries, especially in segments where the waypoints are sparse and GPS coverage is poor. Vision processing is the primary means of accomplishing this task, since roads have a relatively consistent color and texture. Odin is outfitted with two forward-looking monocular cameras, as described above, to achieve a 90 degree field of view. Using visual information gathered from multiple points in front of the vehicle, a threshold operation is performed to filter out non-road particles. The HSL (hue, saturation, and luminance) color space is used for its resistance to changes in lighting conditions. At this point the color image is converted to a binary image to reduce the amount of processing needed for the remaining steps.

A number of particle analysis operations are executed on the binary image to remove false positives and to more clearly define the detected road. These operations remove particles if they are outside a certain size range, occur above the horizon, or do not match the overall geometry of the road. Finally, any holes inside the remaining particles, such as those caused by lines, road markings, or objects on the road, are filled. Once the road has been identified based on its size and geometry, a map of the road coverage is created. This road map is averaged over several frames to minimize the effects of spurious noise.

A secondary system uses two SICK laser rangefinders to supplement the visual road coverage system, scanning the ground for flat drivable areas. Rapid changes in range can be attributed to a discontinuity in the road surface, such as road edges, curbs, potholes and obstacles.

The operations performed by the visual road coverage algorithm remove any markings on the road. A separate algorithm identifies and outputs the location of the visible lane markings. This routine primarily uses intensity to distinguish the bright lane boundaries from the darker surrounding areas of the road surface. Edge detection is applied to the results of the intensity operation, separating out the lines and edges of the road. Finally, the position of each lane is found by fitting the strongest lines on the road to a curve through a Hough transform [Duda, 1972].

3.2.4 Object Classification

The accurate identification and classification of objects is one of the most fundamental and difficult requirements of the Urban Challenge. The vision system and the laser rangefinders each have advantages and disadvantages for classification. The IBEO rangefinders can determine the location of an object to sub-meter accuracy, but they have poor classification capabilities. Vision-based methods can result in accurate classification, but they are computationally intensive, and they have a limited horizontal field of view and range.

Objects are classified into one of two categories: static objects that will not be in motion, and dynamic objects that are in motion or could be in motion. Dynamic objects on the course are expected to be either manned or unmanned vehicles. The core of the classification module, shown in Figure 6, is the IBEO laser rangefinders. The A0 and XT Fusion rangefinders cover almost the entire area around Odin, and objects can be detected using the software available on the IBEO ECUs. Since there can be some variation in each return, an initial filter is applied to remove objects that appear to jump back and forth or to blink in and out of existence. The positions of the remaining objects are checked against the road coverage map of nearby roads; anything not on or close to a known road segment is discarded.

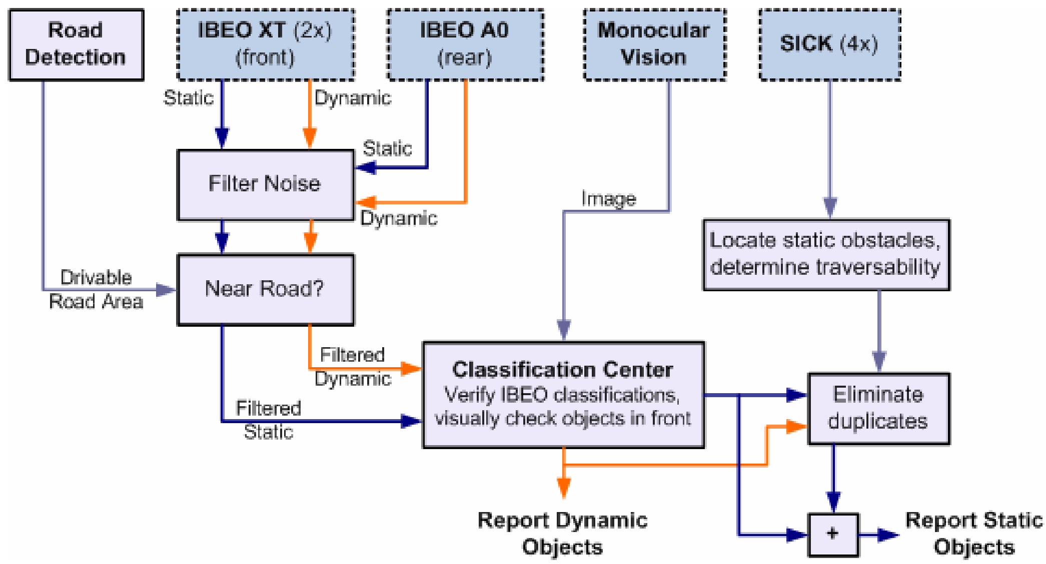


Figure 6: The flowchart for the Object Classification module, which is responsible for determining the location and classification for all perceived objects.

Once these detected objects are sufficiently filtered, their locations and characteristics are passed to a classification center for verification. Through testing, the IBEOs have proven to be accurate in classifying moving objects, and it is assumed that all large moving objects are vehicles. It is also important for the system to detect stationary vehicles in front of Odin for situations such as intersection queuing and precedence. Static and dynamic objects in Odin's path are checked using monocular image processing. The real-world locations of these objects obtained from the IBEOs are converted to regions of interest in the image through a perspective transformation. These regions are examined through vision processing for features common to cars such as tail lights and tires. Because the entire image is never examined, the high resolution of the cameras does not greatly increase processing time. This image processing technique is also used to improve the detection and classification of moving objects. Finally, the SICK rangefinders attempt to locate any positive or negative static obstacles in the otherwise smooth road unseen by the IBEOs.

To deal with conflicting classifications (i.e., when the vision and rangefinder results disagree) a certainty value is determined for each filtered object. The IBEO bases this value on the size, shape, and velocity of the object, while the visual classification bases this value on the number of features that identify the object as a car. The visual certainty value must be greater than that of the IBEO to override the classification, and the overall value must exceed a threshold to identify the object as a vehicle.

Once an object has been classified as a vehicle, it is monitored by the Dynamic Obstacle Predictor, which predicts likely paths for each vehicle based on traffic laws, road data, surrounding obstacles, and the object's motion history. These predictions, shown in Figure 7 are used by Driving Behaviors to classify situations and Motion Planning for obstacle avoidance.

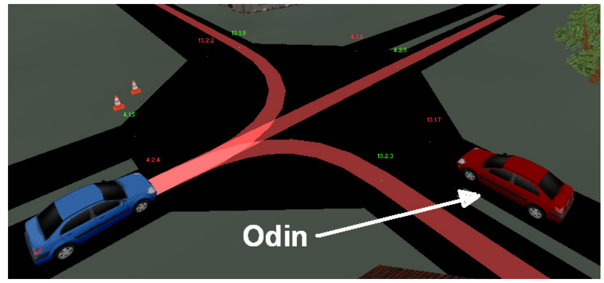


Figure 7: Likely paths generated by the Dynamic Obstacle Predictor for a car detected by Odin.

3.2.5 Localization

The urban environment presents challenges with regard to localization. Urban environments are characterized by tall buildings and other structures that can obstruct direct view of GPS satellites and can reflect GPS signals. In this situation, GPS-based localization may be inaccurate or completely unavailable. To maintain an accurate position estimate, GPS-based localization must be supplemented with inertial, odometric, or other reference data.

Odin has been equipped with an integrated GPS/INS system that will provide a smooth position solution, even during GPS outages. However, after one full minute with no GPS observations, the position uncertainty of the system will increase from 0.1 meters to 2.87 meters RMS. To minimize this drift, an Extended Kalman Filter (EKF) has been developed that combines wheel speed and steering angle measurements with the solution provided by the NovAtel system. The EKF provides an optimal combination of the INS measurements and the vehicle odometry.

The system model for Odin is nonlinear; therefore a linearized version of the Kalman Filter is needed for optimal estimation. The EKF provides the additional benefit of supporting long missions where the nominal trajectory of the vehicle is not available due to the fact that trajectory error estimates are used to update the reference trajectory [Kelly, 1994].

The EKF also allows measurements to be added to the position solution asynchronously. This allows inclusion of measurements taken much less frequently than the INS or odometry. An important reference measurement that is included in this way is the position of stop lines. The rules require that vehicles be able to detect stop lines at intersections and stop within 1 meter of the painted line. These lines can essentially be treated as a high fidelity position measurement. Therefore, no matter how far the position solution drifts from the true vehicle position, the vehicle will always be able to accurately localize itself when it comes to an intersection.

Another problem that arises when dealing with long periods of poor GPS coverage is a phenomenon known as GPS "pop". GPS pop occurs when a vehicle transitions from traveling in an area with poor GPS coverage to an area with accurate GPS coverage. When Odin is moving through an environment without GPS observations, it is relying on inertial and odometric measurements to determine its position. These types of measurements characteristically drift as a function of time and distance traveled. When accurate GPS observations become available again, the position solution suddenly jumps from the purely inertial or odometric solution to the true position of the vehicle. An example of this phenomenon based on data collected while driving through an urban area is shown in Figure 8. Position jumps such as this are problematic for other perception algorithms that rely on the localization solution to build maps or track obstacles. Effectively, this "pop" corrupts all of the data stored in the map. Filtering the INS solution with vehicle odometry measurements can reduce the magnitude of the position jumps, but it cannot eliminate them. Therefore, to avoid this problem, Odin's localization software computes a de-coupled position solution based solely on inertial and odometric measurements. This position solution is referred to as the local position of the vehicle. The vehicle uses the local position solution to determine its position relative to its environment. The global position solution is still used for RNDF-based navigation by higher level decision making components such as the Route Planner and Driving Behaviors modules. This de-coupled solution ensures that the position solution being used by perception components remains continuous with no inexplicable jumps that will corrupt obstacle or map data.

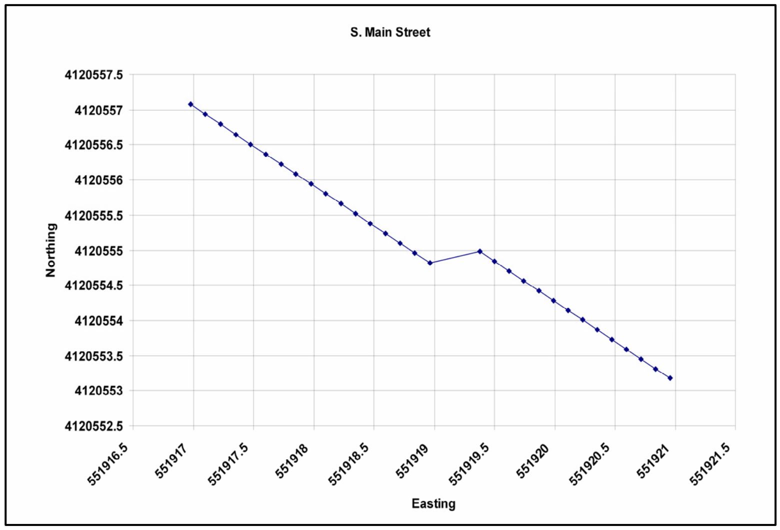


Figure 8: An example of the phenomenon referred to as GPS "pop".

3.3 Planning

Decision making for Odin is handled by a suite of custom developed software. Each of the major components is presented in sequence, from the top down.

3.3.1 Route Planning

The Route Planner component is the coarsest level of decision planning on Odin. It only determines which road segments should be traveled to complete a mission, with decisions such as travel velocity, when to change lanes or how to navigate zones being handled by lower level components. The Route Planner uses a-priori information such as the road network and speed limits specified by the RNDF and MDF respectively as well as information gathered during mission runs. The gathered information includes actual distance between waypoints, any blockages present in a segment, and the speed of traffic. After processing, the Route Planner outputs a series of waypoint exits to travel to each checkpoint in the mission.

By only considering exit waypoints, it is easy to formulate the Route Planner as a graph search problem. The Route Planner on Odin uses the A* graph search method [Hart, 1968] using a time based heuristic to plan the roads traveled. While the A* search algorithm guarantees an optimal solution, it depends on the validity of the data used in the search. The time estimate used during the search assumes that the vehicle is able to travel at the specified segment speed limits, and it uses predefined estimates of the time for typical events, such as the time for a left turn to be clear or the time to traverse a stop line intersection.

The Route Planner also can respond to dynamic situations such as traffic speed or blockages. When blockages are detected and reported to the Route Planner, it marks the blockage in its internal representation of the RNDF and it solves for a new route. A blockage is only stored for 30 minutes to allow for changing course conditions. If a new solution cannot be found when re-planning, previous blockages are removed in order of age until a solution is found.

When calculating the estimate of travel time, the Route Planner uses current traffic speed rather than the speed limit if the traffic speed is available. Accounting for traffic speed allows Odin to take alternate routes if the most direct route is occupied by a slow moving vehicle. Traffic speed is also stored in short term memory to prevent Odin from immediately returning to the same slow road segment. It also prevents Odin from incorrectly penalizing the segment after traffic patterns have changed.

3.3.2 Driving Behaviors

Driving Behaviors is responsible for obeying the rules of the road, producing three outputs. First, Driving Behaviors must produce a behavior profile which takes into account high-level information and defines the general motion of the vehicle to Motion Planning in both roads and zones. Second, the turn signals and horn must be controlled to appropriately signal the intent of the vehicle. Finally, in the event of a road-block, a new set of directions must be requested from the Route Planner.

Due to the number of goals and the variety of inputs that Driving Behaviors must take into account to determine these outputs, a Behavior-Based Paradigm was implemented for this module. The unpredictable nature of an urban environment calls for a robust, vertical decomposition of behaviors. Other advantages of using a Behavior-Based Paradigm for this application include modularity, the ability to test incrementally, and graceful degradation [Murphy, 2000].

As with any Behavior-Based architecture, implementation details are extremely important and can lead to drastically different emergent behaviors. Since no centralized planning modules are used and control is shared amongst a variety of perception-action units, or behaviors, coordination becomes paramount. In the Urban Challenge environment, the problem of action selection in the case of conflicting desires is of particular interest. For example the desire to drive in the right lane due to an upcoming right turn must take precedence over the desire to drive in the left lane due to a slow moving vehicle. The desire to remain stopped at an intersection because of progression order despite a break in traffic is another example. Many comparative studies of various Action Selection Mechanisms (ASM's) such as [Maes, 1991], [Tyrell, 1993], [Bryson, 2000], and [Avila-Garcia, 2002] illustrate the respective merits and drawbacks of different approaches. The majority of ASM's can be broken into two main groups, *arbitration* or *command fusion*, as defined in [Pirjanian, 2000]. Examples of *arbitration* mechanisms include Subsumption Architecture [Brooks, 1986], and Winner-Takes-All strategies such as activation networks [Maes, 1989]. Examples of *command fusion* mechanisms include Potential Field Methods [Arkin, 1987], and the Payton-Rosenblatt voting approach [Rosenblatt, 1995].

Because of the inflexible nature of driving in an urban environment, it is clear that an *arbitration* method of action selection is most appropriate for the Driving Behaviors module. In the above example of choosing the appropriate lane to drive in, driving with two wheels in each lane is not an acceptable solution. Based on the activation energy, only one behavior should be selected and one travel lane sent to Motion Planning. Therefore, a modified Winner-Takes-All mechanism was chosen. The overall architecture of Driving Behaviors is shown in Figure 9. The selected Action Selection Mechanism operates within the Behavior Integrator. This approach is considered a modified Winner-Takes-All approach because all behaviors are broken down into one of three categories: Target Point Drivers, Speed Drivers, and Lane Drivers. This allows for different behaviors, depending on their focus, to all be 'winners'. The Behavior Integrator is therefore responsible for ensuring that there is a 'winner' from each category so a full behavior profile can be generated at any time. This structure also allows for greater modularity and specialization amongst behaviors, which is especially useful in the complexity of the Urban Challenge environment.

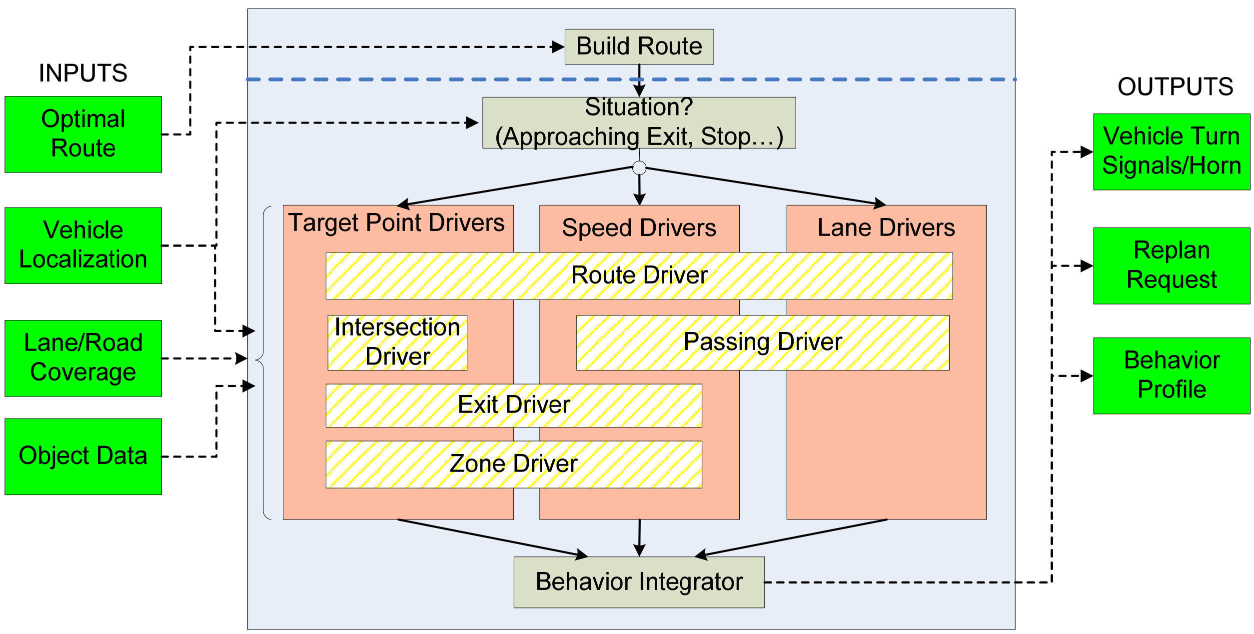


Figure 9: Overall architecture and flow diagram of the Behavior-Based, Winner-Takes-All Driving Behaviors implementation.

The first stage of Driving Behaviors, Build Route, takes the set of directions defined by the Route Planner and builds a more detailed route to follow. This is composed of the exact series of RNDF-defined waypoints that the vehicle would need to follow to reach the required checkpoints in the proper order assuming no other vehicles or obstacles were present. The Build Route module also infers other information from the RNDF and determines the appropriate start point for the mission. For example, the number of forward/oncoming lanes along the route is determined along with the exits to watch at any given intersection. The Build Route module is separated in Figure 9 by a dashed line to indicate that it does not continually run. It is the most time-consuming and processor-expensive part of Driving Behaviors, so it will only run when the

Route Planner sends a new set of directions. This will occur at the beginning of each mission, and in the case that an impassable roadblock is found.

The modules that run continuously within Driving Behaviors are the behaviors themselves. At any given moment, a selection of individual behaviors is chosen to run based on the current situation. This situation is a function of both the goals of the vehicle and the current environment. Each behavior, running in parallel with other behaviors, represents some higher level desire and produces at least one of three standard outputs that compose the behavior profile.

The three standard outputs for any given behavior are Target Points, a Desired Speed, or a Desired Lane. As shown in Figure 9, any single behavior, or driver, can output any combination of these standard outputs. For example, the Passing Driver is both a Speed Driver and a Lane Driver, whereas the Exit Driver is a Speed Driver and a Target Point Driver. The Winner-Takes-All mechanism implemented in the Behavior Integrator selects the appropriate behavior based on an urgency value. For the correct emergent behavior to be produced, it is important to control the urgencies set by individual behaviors carefully. Fine tuning of behavioral urgencies is performed through extensive testing both in simulation and on the vehicle itself.

Ensuring that the behavior profile is capable of dictating the proper overall behavior of the vehicle in every Urban Challenge situation is another important design consideration. Examples include dictating a controlled passing maneuver, commanding the proper lane for an exit or checkpoint, navigating and parking in a zone, executing U-turns, and following right-of-way rules at an intersection without excess delay. The full behavior profile therefore consists of a Desired Velocity, a Desired Lane, a Direction Indicator, which dictates forward or reverse travel, and three Target Points. Each Target Point consists of an X and Y location within the local frame, a set of Behavior Flags, an optional heading, and the Lane ID of that Target Point. The Behavior Flags include a Stop Flag, which indicates to Motion Planning that the vehicle must come to a complete stop at that Target Point and remain stopped until a new set of Target Points are sent with the Stop Flag removed. In the case of a road, the Target Points are used to indicate the general direction of travel; it is assumed that lower level control issues such as lane maintenance, especially in the case of sparse waypoints, are handled within Motion Planning. Furthermore, in zones, Target Points will be used to guide the vehicle into common travel areas and to avoid obvious trap situations, but lower-level obstacle avoidance will again be handled within Motion Planning.

3.3.3 Motion Planning

Motion Planning is the decision making layer between Driving Behaviors and the Vehicle Interface that converts target points into a series of vehicle commands. Motion Planning generates a navigation strategy to safely achieve these set goal points using the software flow diagram shown in Figure 10. The two main subdivisions are lanes and zones. Lane navigation requires the vehicle to maintain strict boundaries and conform to the motion other vehicles. If static obstacles are present and an achievable trajectory exists, Motion Planning will navigate around them. Zones, on the other hand, are much less structured and require a balance of speed and steering based obstacle avoidance.

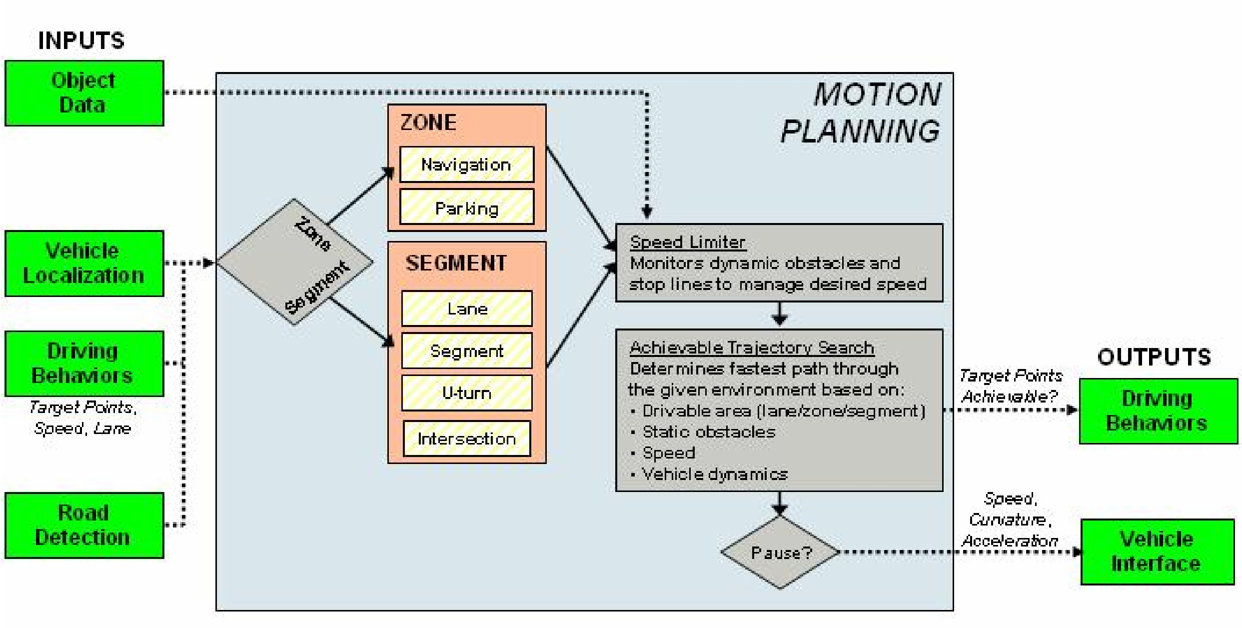


Figure 10: Software Flow Diagram of the Motion Planning Component

The core of the Motion Planning process is an achievable trajectory search that simulates future motion to determine a safe and fast path through the sensed environment. The search starts with the current vehicle state and uses an A* search to evaluate sequences of possible future motion commands. The search speed is improved by only using a finite set of possible actions that have pre-computed simulation results [Lacaze, 1998]. The search favors motion commands that will reach target points in minimum time while avoiding obstacles and keeping lateral accelerations within safe limits. By constraining the set of available motions, the same algorithm can be used for driving in a lane, navigating through a zone, and positioning a vehicle for parking.

To assist the achievable trajectory search, the speed limiter monitors dynamic obstacles and stop line positions. Dynamic obstacles are evaluated using a robust algorithm called Velocity Obstacles [Firorini, 2001]. This algorithm reduces the number of searchable trajectories by outputting a velocity space that yields collision-free motion. Similar to the behaviors of a human driver, one of three collision avoidance maneuvers can be chosen for each obstacle: moving in front, moving behind, or diverging. The chosen behavior is dependent on the trajectory of the dynamic obstacle. Figure 11 shows the three different maneuvers in the example of another car merging into Odin's lane. Velocity Obstacles operates on relative velocities, desensitizing it to object range and rate of detection. A second speed constraint is used to reduce speed when

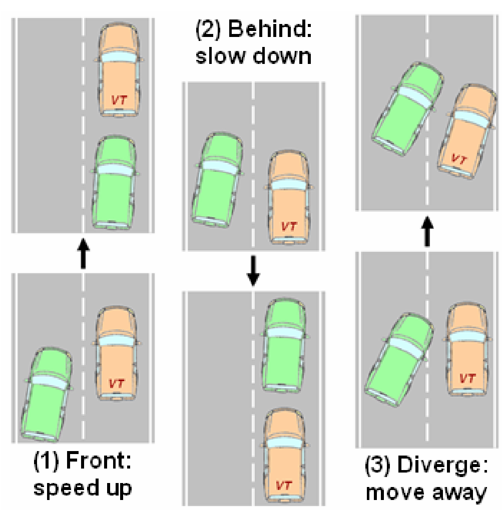


Figure 11: The three avoidance maneuvers considered by Velocity Obstacles

approaching a stop line. A desired speed is output based on the distance away from the stop line and the current speed of the vehicle. Monitoring these values allows Motion Planning to command a smooth, controlled stop.

3.3.4 Vehicle Interface

The main role of the Vehicle Interface component is to interpret the generic motion profile messages from Motion Planning and output vehicle-specific throttle, brake, steering, and shifting signals. This requires closed-loop speed and steering control as well as transmission actuation. By accepting generic motion commands, any updates to the vehicle-specific hardware or software will be transparent to the higher level decision making components. In addition to simply following the motion profile commands, the Vehicle Interface implements simple rollover prevention by modifying any potentially unsafe motion profile commands. It is also responsible for handling the vehicle mode, actuating the audible and visual indicators, and monitoring the vehicle's CAN bus for the vehicle state information, such as odometry, needed by Localization.

To provide a high level of software stability, the Vehicle Interface runs on a National Instrument's Compact RIO controller. The messaging and higher level controls take place on the real-time processor of the RIO, while the lower level closed-loop control of speed and curvature occur at very high speeds (between 5 and 10 kHz) on the computer's FPGA (Field Programmable Gate Array) modules. The component uses six separate threads to perform all of its tasks: JAUS messaging, motion profile handling, vehicle mode handling, speed control, steering control, and vehicle data bus reading. This prevents slower processes such as JAUS messaging, which only runs each time a message is received, from holding back the more time critical processes, such as speed and steering control. Thus, the Vehicle Interface can perform all required tasks at separate speeds to actuate each of the systems needed to control the vehicle.

3.4 Base Platform

The base platform for Odin is a 2005 Ford Escape Hybrid that has been outfitted with drive-by-wire controls, power systems, sensor arrays, and computing systems.

3.4.1 Vehicle Selection & Conversion

Team ***V***ictor***T***ango has selected a 2005 Ford Escape Hybrid as the platform for Odin. The Hybrid Escape contains several features advantageous for use as an autonomous vehicle. Throttle and shift controls are natively drive-by-wire and the steering system is electrically, rather than hydraulically, assisted. These features help simplify the resulting conversion to computer control. The vehicle also has a powerful on-board generator and high-voltage battery system. This system is capable of providing excess electrical power for computing and sensing systems while being charged and managed by the factory control system. The vehicle also provides 0.75 m^3 of cargo space to mount electronics, as shown in Figure 12. At the same time, the vehicle is small enough at 1.8m, wide by 4.4m long to maintain good maneuverability and dynamic performance. Ford Motor Company joined team ***V***ictor***T***ango and provided two base vehicle platforms, technical support and access to proprietary information, including vehicle CAN bus protocols. All of these factors made the Hybrid Escape an excellent choice for the base vehicle platform.

The throttle, shifting, and steering systems were converted to drive-by-wire by tapping into the stock control inputs and simulating the signals to drive the factory systems. No actuators were added to the vehicle to control these functions. These systems are fitted with automatic relays that will disable the systems for human control or emergency situations. The brake was fitted with a pedal-mounted linear actuator. A separate, spring-actuated failsafe emergency brake system has also been fitted to Odin.

A closed-loop, feed-forward speed control algorithm that directly compensates for inclines, maintains the desired vehicle velocity over a wide range of terrain. The vehicle is capable of accelerations of 2.5 m/s^2, decelerations of 7 m/s^2, and velocity tracking accuracies of 0.1 m/s. A curvature-based steering control algorithm uses a simple dynamic bicycle model to account for vehicle dynamic effects on curvature control. This model corrects for the effects of velocity and tire slip in order to increase the accuracy of curvature following over a strictly theoretical model.

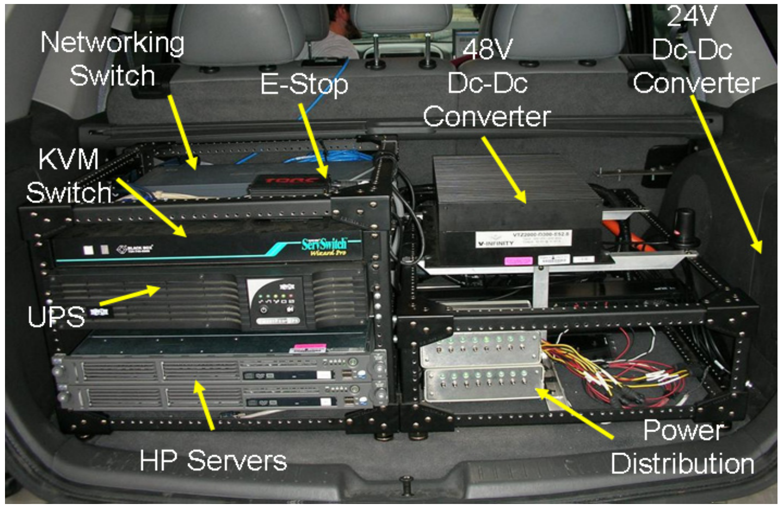


Figure 12: Odin's electronics rack showing computing and power systems. Note that the vehicle retains full passenger capability.

The computing and sensor systems are powered by a 2kW 48V DC-DC converter attached to the high-voltage hybrid power system on the Escape. Eighty percent of this power is fed to a Tripp-Lite Uninterruptible Power Supply to provide redundant AC power to the computing systems. Power for sensors is provided by a secondary DC-DC converter to provide 400 Watts of clean 24V power. The total available capacity of the power system is 2kW, of which approximately 40% is currently being used. All of the major power and computing systems are mounted in the rear cargo compartment of Odin. Cooling is provided by ducting the stock air-conditioning system to the rear.

3.4.2 Computing Systems

The two primary objectives in designing the computer system were to reduce hardware complexity while still achieving the benefits of high performance and process isolation inherent in a machine cluster. Additionally the ability to easily reallocate computing resources according to the demand of the individual software components was desired. The use of virtual machines running on a single computer offered an ideal solution. By using virtual machines, one high performance machine has been efficiently partitioned and has precise hardware allocation control.

Using virtual machines does introduce some technical challenges in the computing system architecture. For example, high-speed devices requiring DMA transfers such as firewire, high-speed USB, or video capture devices are generally not supported under virtualization.

Additionally, the software development system used for machine vision development is only compatible with Microsoft's Windows operating system. To overcome issues of hardware compatibility and performance, the team decided to use a hybrid system consisting of a Linux para-virtualized machine cluster and a single high performance Windows XP machine. Both machines are HP Proliant DL140 rackmount servers with a 2 Ghz quad core CPU and 4GB of RAM. They are connected to a gigabit ethernet switch, which serves as the backbone for the vehicle communications system. Most of the devices, including the IBEO laser rangefinders, the Vehicle Interface, and the SICK LMS, either operate natively over the network or, in the case of the SICKs, employ a Serial-to-Ethernet bridge. Inter-process communication is handled over the network through the JAUS architecture. The Linux computer also provides network services such as DHCP, NIS, and DNS for ease of development and NTP for time synchronization. Additionally, the Linux computer serves as a file store for the entire vehicle system.

While the Linux computer serves as the main processing system for network abstracted sensors, the Windows XP computer is primarily used to run computer vision algorithms and interface with hardware that may not be accessible through the virtual machines. Both computers are connected over a KVM switch to a console in the front of the vehicle and every computer, including the virtual machines, has a graphical terminal that is accessible over VNC, allowing several users to simultaneously develop, monitor, and debug the system.

4. Results & Performance

Groups that design and implement a vehicle system are often more focused on how that component will work rather than how it might fail. To provide a system of checks and balances, team ***V***ictor***T***ango has formed an independent test team. The primary responsibility of the test team is to develop and conduct tests on the vehicle that are meant to push systems to the limits of their performance and expose their weaknesses. By doing so early in development, the reliability of the vehicle is improved.

4.1 General Test Team Methodology

Two primary methods are used to test vehicle components and software. A simulator has been developed that allows many functions of the software algorithms to be quickly tested. Vehicle systems and software are also tested on Odin in a variety of real driving tests. Both methods are used to validate successful operation of all vehicle components.

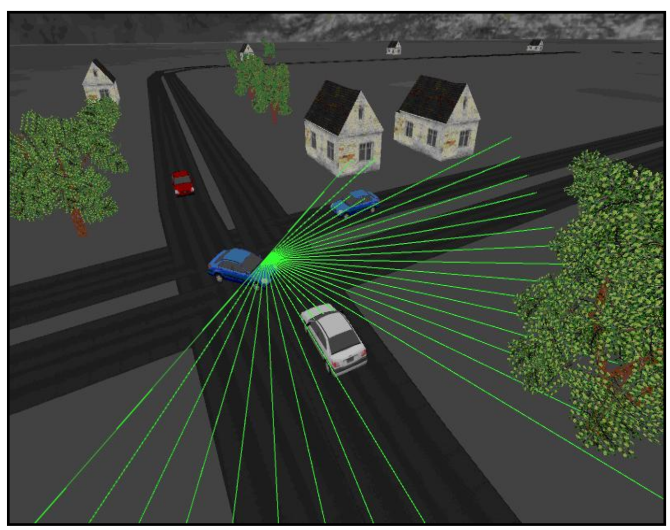


Figure 13: Screen capture from TORC simulator

The simulator provides a virtual environment for testing software algorithms. Rather than running in a real environment where untested software could pose safety concerns and cause an accident that damages the vehicle and sensors, testing in the virtual environment is free from most real world consequences. The simulator also allows for testing on virtually any course configuration, providing the team with the ability to test in a broad range of situations. Figure 13 shows a screen capture from the simulator.

On-road testing has been performed to evaluate sensor performance, software behaviors, and navigational accuracy and stability. A wide variety of test scenarios encompassing the behaviors anticipated for the Urban Challenge event have been developed and are currently in use.

During testing, data files consisting of vehicle performance parameters, sensor data, and software commands are logged. These data sets can be replayed through the simulator to analyze vehicle behaviors and performance criteria. Comparison of previous logged tests permits performance evaluation for software development.

4.2 Component Test Results

Although Odin is still under development at the time of this writing, extensive testing has been performed to validate the design decisions and verify system performance. Odin's full capabilities will be displayed at the site visit in June, at the NQE in October, and at the Urban Challenge Final Event (UFE) in November, 2007.

4.2.1 Perception Testing

The team has demonstrated visual detection of a variety of roads and lane markings, and example of which is shown in Figure 14. The road width has been determined to within 1 meter accuracy at 15 meters out, and 2 meter accuracy at 35 meters out. Although some of the edges are rough and inaccurate, this noise is minimized by averaging successive frames of the coverage maps and taking into account objects obscuring the side of the road. The lane markings are currently approximated with straight lines that are accurate to within 0.25 meters near the vehicle, and within 1 meter at 15 meters out. The lines appear to intersect in Figure 14 because their locations have not yet been transformed out of the perspective view. A more accurate determination of the overall lane width can be determined by assuming the width will be constant for a single lane, and by fitting curved lines through the Hough Transform.

Figure 14: Example road coverage overlay. The green area is the road detected by the module, and the red lines define the bounds for the current lane of travel.

SICK laser rangefinders are also used to improve the road coverage map. For the range that the SICKs cover they are capable of sub-30-cm accuracy. When oriented 19 degrees below horizontal the scans contact flat ground 6 meters ahead of the vehicle, allowing the SICK to accurately measure the road width to within 0.1 meters. If oriented 9 degrees below horizontal the range is increased to almost 12 meters. However, the error for this orientation increases to between 0.2 and 1 meter depending on the accuracy of the road edges.

The IBEO has proven to be accurate in detecting the presence of dynamic objects. Within 75 meters it can locate moving objects with sub-meter accuracy and measure velocities with an average error of 0.15 miles per hour at 20 miles per hour. However, as noted earlier, the classification of static objects has proven inconsistent. Within 30 meters of Odin the classification of a static vehicle is only 25-50% accurate due to the large number of returns distorting the profile. This accuracy actually improves to 75-80% between the 30 and 60 meter range as fewer scan planes intersect the objects. Basic filtering has improved the short range capabilities of the classification module to detect approximately 85% of vehicles within 30 meters, but this approach does not improve distant classification due to the limited number of scan returns.

Visual detection of preceding vehicles has also been tested and verified. Using a high resolution image, unlit tail lights are detected at up to 30 meters, as shown in Figure 15. The lights can be detected at a further distance, but as a result, there is a larger number of false positives even after particle analysis filtering. Coupling the system with the IBEO's detection capabilities allows for a more precise description of the probable position of vehicle tail lights, which in turn will remove these false positives. It also allows the system to function in a wider variety of lighting conditions.

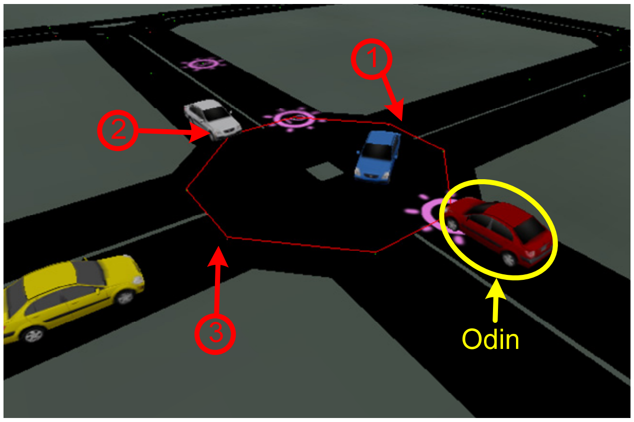
Figure 15: A vehicle detected at a distance of 30 meters.

4.2.2 Decision Making Testing

Decision making in complex situations is the focal problem presented by the DARPA Urban Challenge. To improve robustness and maximize testing efficiency, isolated testing of the decision making process is essential. By utilizing the simulator, action selection in such complex situations as a 4-way intersection can be thoroughly validated.

As shown in Figure 16, the 4-way intersection scenario is broken down into seven cases based on the relative position of other vehicles in the intersection. The relative position of each vehicle is classified as either arriving, stopped, or entering the intersection from the given stop point. From these relative positions, an expected result can be determined by following typical rules of the road. In Figure 16, the 5th situation is shown, in which upon Odin's arrival, one vehicle is entering the intersection, another is already stopped, and a third is still arriving at the stop point. Given this situation, it is expected that Odin will cross the intersection 3rd.

After performing these tests, action-selection in 4-way and 3-way stops was verified to have a 100% success rate. An incorrect decision in an urban driving environment can easily present a safety threat, possibility for damage, and result in failure of the mission. All decision making components will therefore be tested exhaustively in this two-stage manner both isolated (in simulation) and integrated with perception (in the real-world).



Intersection Position			Expected Result
1	2	3	
A	*A*	*A*	1st
A	*A*	*S*	2nd
A	*A*	*E*	2nd
A	*S*	*S*	3rd
A	*S*	*E*	3rd
S	*S*	*E*	4th
S	*S*	*S*	4th

A = Vehicle Arriving, S = Vehicle Stopped,
E = Vehicle Entering Intersection

Figure 16: Intersection test setup with a matrix of validated situations.

4.2.3 Base Platform Testing

To enable safe obstacle avoidance, the dynamic rollover model of Odin was verified by data collection during manual operation of the vehicle. A professional stunt driver maneuvered the vehicle through a simulated obstacle course. This data, as shown by the red line in Figure 17, shows the dynamic limits of the vehicle and ensures that the operating conditions allowed by the dynamic model are safe for autonomous operation. The test also verified the performance of the steering actuator, as the autonomous system is able to produce steering rates of up to 520 degrees/s, which easily exceeds the maximum rate of the human driver even in extreme situations.

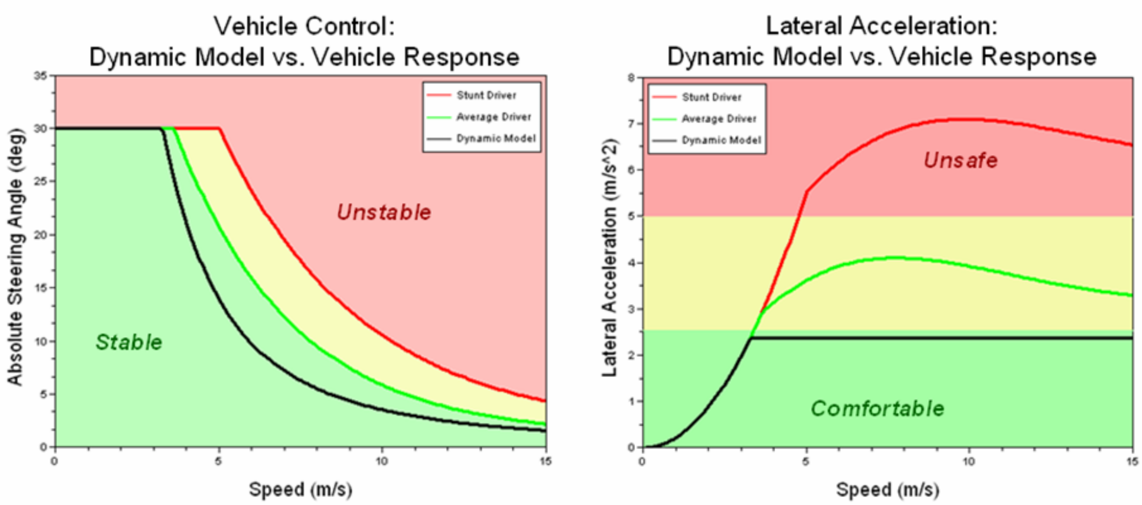


Figure 17: Vehicle dynamic stability results. Red line shows the limit of Odin's dynamic capabilities. The black line is the max allowable autonomous performance.

5. Conclusions

Team ***V***ictor***T***ango has made significant progress since the Urban Challenge project began, and is continuing to work on many aspects of the problem. Most importantly, we have invested significant effort to organize the system architecture, and to formulate a well-thought-out development plan and the software elements needed to be successful in one of the most complex and exciting challenges of our generation.

As part of this foundation, ***V**ictor**T**ango* has created an ideal base vehicle platform for autonomous urban navigation. Odin is meticulously packaged with ample on-board power available directly from the stock hybrid power train, and the drive-by-wire conversion taps into the stock control signals for throttle, shift and steering yielding a safe, convenient and reliable base platform. The design is revolutionary in the sense that autonomy is being built in, rather than added on to, a commercial passenger vehicle.

This same type of solid foundation extends into all the other areas of our Urban Challenge effort, for example, in developing tools such as JAUS software toolkit and a custom Urban Challenge Software Simulator. The exclusive and extensive use of JAUS provides a level of modularity and software portability that would be difficult to achieve without such a messaging architecture. This implementation of JAUS is revolutionary in that it extends the standard's typical teleoperation message set to enable the most sophisticated levels of autonomous operation. The custom Urban Challenge Software simulator facilitates testing in a variety of software-only and hardware-in-the-loop situations; such flexibility in testing streamlines and accelerates our development effort greatly.

From the start, perception software has been validated on the real vehicle, decoupled from planning software developed in simulation, allowing algorithms to be refined in the most efficient manner. Presently, with the individual perception and planning components developed and validated independently, testing and refinement of Basic Navigation and Basic Traffic behaviors in real-world in-vehicle testing is proceeding rapidly. As a result, ***V**ictor**T**ango* is well positioned for the integration of the Advanced Navigation and Traffic behaviors under development now.

Team ***V**ictor**T**ango* is on schedule in every phase of work to accomplish all of the Basic Navigation, Basic Traffic, Advanced Navigation and Advanced Traffic behaviors in the DARPA Urban Challenge. Currently, Odin undergoes almost daily testing, with the results demonstrating its ability to detect and classify roads and obstacles, make proper driving decisions, and safely execute those decisions. In the next few weeks, final physical validation will be complete for all 16 Basic Navigation and Basic Traffic behaviors required at the Milestone 2 visit in late June.

In conclusion, team ***V**ictor**T**ango* has developed a solid base platform and produced a suite of perception and control software that will be capable of completing the most complex fully-autonomous ground vehicle challenge in history. Current test results indicate that Odin demonstrates the necessary robustness, intelligence, and adaptability to operate in a dynamic, unpredictable environment without human intervention. By utilizing a solid base vehicle, advanced sensor fusion techniques, and a unique implementation of a Hybrid Deliberative-Reactive software architecture, team ***V**ictor**T**ango* is breaking new ground in the field of autonomous ground vehicles.

References

1. Arkin, R. C. (1987). “Motor schema based navigation for a mobile robot: An approach to programming by behavior”. In *Robotics and Automation*. Proceedings. 1987 IEEE International Conference on, IEEE.
2. Avila-Garcıa, O., Hafner, E., and Canamero, L. (2002). “Relating Behavior Selection Architectures to Environmental Complexity.” In *Proc. Seventh Intl. Conf. on Simulation of Adaptive Behavior*. Cambridge, MA: MIT Press.
3. Brooks, R. A. (1986). “A Robust Layered Control System for a Mobile Robot.” In *IEEE Journal of Robotics and Automation,* Vol. 2 (1) :14–23.
4. Bryson, J. (2000). “Hierarchy and Sequence vs. Full Parallelism in Action Selection.” In J.A. Meyer, A. Berthoz, D. Floreano, H. Roitblat, and S.W. Wilson, eds., *Proc. Sixth Intl. Conf. on Simulation of Adaptive Behavior*, 147–156. Cambridge, MA: MIT Press.
5. Duda, R. O. and P. E. Hart (1972). “Use of the Hough Transform to Detect Lines and Curves in Pictures.” In Commun. ACM Vol 15(1), pp. 11-15
6. Fiorini, Paolo and Zvi Shiller (2001). *Motion Planning in Dynamic Environments using Velocity Obstacles*. Proc. International Conference of Robotics and Automation. 2001. V4. pp. 3716-3721.
7. Hart, P. E.; Nilsson, N. J.; Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". IEEE Transactions on Systems Science and Cybernetics SSC4 (2): pp. 100–107.
8. Kelly, A. J. (1994). “A 3D State Space Formulation of a Navigation Kalman Filter for Autonomous Vehicles.” CMU Robotics Institute Technical Report CMU-RI-TR-94-19.
9. Lacaze, A., Moscovitz, Y., DeClaris, N., Murphy, K (1998). *Path Planning for Autonomous Vehicles Driving Over Rough Terrain*. Proceedings of the ISIC/CIRA/ISAS Conference. Gaithersburg, MD. September 14-17, 1998.
10. Maes, P. (1991). “A Bottom-Up Mechanism for Behavior Selection in an Artificial Creature.” In J.A. Meyer and S.W. Wilson, eds. *Proc. First Intl. Conf. on Simulation of Adaptive Behavior*, 238–246. Cambridge, MA: MIT Press.
11. Maes, P. (1989). “How To Do the Right Thing.” Technical Report NE 43–836, AI Laboratory, MIT, Cambridge, MA.
12. Murphy, R. R. (2000). *Introduction to AI Robotics*. Cambridge, MA: MIT Press.
13. Pirjanian, P. (2000). “Multiple Objective Behavior-Based Control.” In *Robotics and Autonomous Systems*, Vol. 31(1):53-60.
14. Rosenblatt, J. K. (1995) “DAMN: A Distributed Architecture For Mobile Navigation.” In *Proceedings of the AAAI Spring Symposium on Lessons Learned from Implemented Software Architectures for Physical Agents*, Stanford, CA, AAAI Press.
15. Tyrrell, T. (1993). “The Use of Hierarchies for Action Selection.” *Adaptive Behavior*, 1(4): 387–419.